EXHIBIT AA

EXHIBIT AA
SYS V init Manual Pages

inittab(4)

inittab -- script for init

Description

The file */etc/inittab* controls process dispatching by *init*. The processes most typically dispatched by *init* are daemons.

The *inittab* file is composed of entries that are position dependent and have the following format:

```
id:rstate:action:process
```

Each entry is delimited by a newline, however, a backslash (\) preceding a newline indicates a continuation of the entry. Up to 512 characters per entry are permitted. Comments may be inserted in the ***process*** field using the convention for comments described in **sh**(1). There are no limits (other than maximum entry size) imposed on the number of entries in the *inittab* file. The entry fields are:

id

This is one to four characters used to uniquely identify an entry.

rstate

This defines the run level in which this entry is to be processed. Run-levels effectively correspond to a configuration of processes in the system. That is, each process spawned by *init* is assigned a run level or run levels in which it is allowed to exist. The run levels are represented by a number ranging from **0** through **6**. As an example, if the system is in run level **1**, only those entries having a **1** in the ***rstate*** field are processed. When *init* is requested to change run levels, all processes that do not have an entry in the ***rstate*** field for the target run level are sent the warning signal **SIGTERM** and allowed a 5-second grace period before being forcibly terminated by the kill signal **SIGKILL**. The ***rstate*** field can define multiple run levels for a process by selecting more than one run level in any combination from **0** through **6**. If no run level is specified, then the process is assumed to be valid at all run levels **0** through **6**. There are three other values, **a**, **b** and **c**, which can appear in the ***rstate*** field, even though they are not true run levels. Entries which have these characters in the ***rstate*** field are processed only when an *init* or *telinit* process requests them to be run (regardless of the current run level of the system). See **init**(1M). They differ from run levels in that *init* can never enter run level **a**, **b** or **c**. Also, a request for the execution of any of these processes does not change the current run level. Furthermore, a process started by an **a**, **b** or **c** command is not killed when *init* changes levels. They are killed only if their line in *inittab* is marked **off** in the ***action*** field, their line is deleted entirely from *inittab*, or *init* goes into single-user state.

EXHIBIT AA
SYS V init Manual Pages

action

Key words in this field tell *init* how to treat the process specified in the ***process*** field. The actions recognized by *init* are as follows:

respawn
If the process does not exist, then start the process; do not wait for its termination (continue scanning the *inittab* file), and when the process dies, restart the process. If the process currently exists, do nothing and continue scanning the *inittab* file.

wait
When *init* enters the run level that matches the entry's ***rstate***, start the process and wait for its termination. All subsequent reads of the *inittab* file while *init* is in the same run level cause *init* to ignore this entry.

once
When *init* enters a run level that matches the entry's ***rstate***, start the process, do not wait for its termination. When it dies, do not restart the process. If *init* enters a new run level and the process is still running from a previous run level change, the program is not restarted.

boot
The entry is to be processed the first time *init* goes from single-user to multi-user state after the system is booted. (If *initdefault* is set to **2**, the process runs right after the boot.) *init* starts the process, does not wait for its termination and, when it dies, does not restart the process.

bootwait
The entry is to be processed the first time *init* goes from single-user to multi-user state after the system is booted. (If *initdefault* is set to **2**, the process runs right after the boot.) *init* starts the process, waits for its termination and, when it dies, does not restart the process.

powerfail
Execute the process associated with this entry only when *init* receives a power fail signal, **SIGPWR** [see **signal**(2)].

powerwait
Execute the process associated with this entry only when *init* receives a power fail signal, **SIGPWR**, and wait until it terminates before continuing any processing of *inittab*.

off
If the process associated with this entry is currently running, send the warning signal **SIGTERM** and wait 5 seconds before forcibly terminating the process with the kill signal **SIGKILL**. If the process is nonexistent, ignore the entry.

ondemand

This instruction is really a synonym for the **respawn** action. It is functionally identical to **respawn** but is given a different keyword in order to divorce its association with run levels. This instruction is used only with the **a**, **b** or **c** values described in the ***rstate*** field.

initdefault
An entry with this action is scanned only when *init* is initially invoked. *init* uses this entry, if it exists, to determine which run level to enter initially. It does this by taking the highest run level specified in the ***rstate*** field and using that as its initial state. If the ***rstate*** field is empty, this is interpreted as **0123456** and *init* therefore enters run level **6**. This will cause the system to loop, that is, it will go to firmware and reboot continuously. Additionally, if *init* does not find an *initdefault* entry in *inittab*, it requests an initial run level from the user at reboot time.

sysinit
Entries with this action are scanned only when *init* is initially invoked. Among other things, *sysinit* entries may be used to initialize devices on which *init* might try to ask the run level question. These entries are executed and waited for before continuing.

process
This is a command to be executed. The entire `process` field is prefixed with **exec** and passed to a forked **sh** as **sh -c 'exec *command*'**. For this reason, any legal **sh** syntax can appear in the ***process*** field.

Notices

The **wsinit** command is required to initialize the system console. Do not remove this file, attempt to run it from the command line, or remove the line invoking it from */etc/inittab* or */etc/conf/init.d/kernel*.

Application code should not attempt to modify the */etc/inittab* file during a run-level change, since the *etc/init* program ignores inittab changes then. In particular, modifying the */etc/inittab* file while the system is shutting down will result in minor root file system damage.

Files

/sbin/wsinit

References

exec(2), **init**(1M), **open**(2), **sh**(1), **signal**(2), **ttymon**(1M), **who**(1)

EXHIBIT DD

EXHIBIT DD

Linux ELF Code

```
#ifndef _LINUX_ELF_H
#define _LINUX_ELF_H

#include <linux/types.h>
#include <asm/elf.h>

/* 32-bit ELF base types. */
typedef __u32     Elf32_Addr;
typedef __u16     Elf32_Half;
typedef __u32     Elf32_Off;
typedef __s32     Elf32_Sword;
typedef __u32     Elf32_Word;
                                        <signature: >

/* 64-bit ELF base types. */
typedef __u64     Elf64_Addr;
typedef __u16     Elf64_Half;
typedef __s16     Elf64_SHalf;
typedef __u64     Elf64_Off;
typedef __s32     Elf64_Sword;
typedef __u32     Elf64_Word;
typedef __u64     Elf64_Xword;
typedef __s64     Elf64_Sxword;

/* These constants are for the segment types stored in the image headers */
#define PT_NULL    0
#define PT_LOAD    1
#define PT_DYNAMIC 2
#define PT_INTERP  3
#define PT_NOTE    4
#define PT_SHLIB   5
#define PT_PHDR    6
#define PT_LOPROC  0x70000000
#define PT_HIPROC  0x7fffffff
#define PT_MIPS_REGINFO       0x70000000

/* Flags in the e_flags field of the header */
#define EF_MIPS_NOREORDER 0x00000001
#define EF_MIPS_PIC       0x00000002
#define EF_MIPS_CPIC      0x00000004
#define EF_MIPS_ARCH      0xf0000000

/* These constants define the different elf file types */
#define ET_NONE   0
#define ET_REL    1
#define ET_EXEC   2
#define ET_DYN    3
#define ET_CORE   4
#define ET_LOPROC 0xff00
#define ET_HIPROC 0xffff

/* These constants define the various ELF target machines */
#define EM_NONE  0
#define EM_M32   1
```

EXHIBIT DD

Linux ELF Code

```
#define EM_SPARC 2
#define EM_386   3
#define EM_68K   4
#define EM_88K   5
#define EM_486   6   /* Perhaps disused */
#define EM_860   7

#define EM_MIPS         8     /* MIPS R3000 (officially, big-endian only) */

#define EM_MIPS_RS4_BE 10     /* MIPS R4000 big-endian */

#define EM_PARISC      15     /* HPPA */

#define EM_SPARC32PLUS 18     /* Sun's "v8plus" */

#define EM_PPC          20    /* PowerPC */
#define EM_PPC64       21       /* PowerPC64 */

#define EM_SH           42    /* SuperH */

#define EM_SPARCV9     43     /* SPARC v9 64-bit */

#define EM_IA_64  50    /* HP/Intel IA-64 */

#define EM_X86_64 62    /* AMD x86-64 */

#define EM_S390         22    /* IBM S/390 */

#define EM_CRIS         76      /* Axis Communications 32-bit embedded processor */

/*
 * This is an interim value that we will use until the committee comes
 * up with a final number.
 */
#define EM_ALPHA  0x9026

/*
 * This is the old interim value for S/390 architecture
 */
#define EM_S390_OLD     0xA390

/* This is the info that is needed to parse the dynamic section of the file */
#define DT_NULL         0
#define DT_NEEDED 1
#define DT_PLTRELSZ     2
#define DT_PLTGOT 3
#define DT_HASH         4
#define DT_STRTAB 5
#define DT_SYMTAB 6
#define DT_RELA         7
#define DT_RELASZ 8
#define DT_RELAENT      9
#define DT_STRSZ  10
#define DT_SYMENT 11
#define DT_INIT         12
```

EXHIBIT DD

Linux ELF Code

```
#define DT_FINI          13
#define DT_SONAME 14
#define DT_RPATH  15
#define DT_SYMBOLIC      16
#define DT_REL             17
#define DT_RELSZ  18
#define DT_RELENT 19
#define DT_PLTREL 20
#define DT_DEBUG  21
#define DT_TEXTREL       22
#define DT_JMPREL 23
#define DT_LOPROC 0x70000000
#define DT_HIPROC 0x7fffffff
#define DT_MIPS_RLD_VERSION   0x70000001
#define DT_MIPS_TIME_STAMP    0x70000002
#define DT_MIPS_ICHECKSUM     0x70000003
#define DT_MIPS_IVERSION      0x70000004
#define DT_MIPS_FLAGS         0x70000005
  #define RHF_NONE              0
  #define RHF_HARDWAY           1
  #define RHF_NOTPOT            2
#define DT_MIPS_BASE_ADDRESS  0x70000006
#define DT_MIPS_CONFLICT      0x70000008
#define DT_MIPS_LIBLIST       0x70000009
#define DT_MIPS_LOCAL_GOTNO   0x7000000a
#define DT_MIPS_CONFLICTNO    0x7000000b
#define DT_MIPS_LIBLISTNO     0x70000010
#define DT_MIPS_SYMTABNO      0x70000011
#define DT_MIPS_UNREFEXTNO    0x70000012
#define DT_MIPS_GOTSYM        0x70000013
#define DT_MIPS_HIPAGENO      0x70000014
#define DT_MIPS_RLD_MAP       0x70000016

/* This info is needed when parsing the symbol table */
#define STB_LOCAL  0
#define STB_GLOBAL 1
#define STB_WEAK   2

#define STT_NOTYPE  0
#define STT_OBJECT  1
#define STT_FUNC    2
#define STT_SECTION 3
#define STT_FILE    4

#define ELF32_ST_BIND(x) ((x) >> 4)
#define ELF32_ST_TYPE(x) (((unsigned int) x) & 0xf)

/* Symbolic values for the entries in the auxiliary table
   put on the initial stack */
#define AT_NULL   0     /* end of vector */
#define AT_IGNORE 1     /* entry should be ignored */
#define AT_EXECFD 2     /* file descriptor of program */
#define AT_PHDR   3     /* program headers for program */
#define AT_PHENT  4     /* size of program header entry */
#define AT_PHNUM  5     /* number of program headers */
```

EXHIBIT DD

Linux ELF Code

```
#define AT_PAGESZ 6     /* system page size */
#define AT_BASE   7     /* base address of interpreter */
#define AT_FLAGS  8     /* flags */
#define AT_ENTRY  9     /* entry point of program */
#define AT_NOTELF 10    /* program is not ELF */
#define AT_UID    11    /* real uid */
#define AT_EUID   12    /* effective uid */
#define AT_GID    13    /* real gid */
#define AT_EGID   14    /* effective gid */
#define AT_PLATFORM 15  /* string identifying CPU for optimizations */
#define AT_HWCAP  16    /* arch dependent hints at CPU capabilities */
#define AT_CLKTCK 17    /* frequency at which times() increments */

typedef struct dynamic{
  Elf32_Sword d_tag;
  union{
    Elf32_Sword   d_val;
    Elf32_Addr    d_ptr;
  } d_un;
} Elf32_Dyn;

typedef struct {
  Elf64_Sxword d_tag;         /* entry tag value */
  union {
    Elf64_Xword d_val;
    Elf64_Addr d_ptr;
  } d_un;
} Elf64_Dyn;

/* The following are used with relocations */
#define ELF32_R_SYM(x) ((x) >> 8)
#define ELF32_R_TYPE(x) ((x) & 0xff)

#define R_386_NONE      0
#define R_386_32  1
#define R_386_PC32      2
#define R_386_GOT32     3
#define R_386_PLT32     4
#define R_386_COPY      5
#define R_386_GLOB_DAT  6
#define R_386_JMP_SLOT  7
#define R_386_RELATIVE  8
#define R_386_GOTOFF    9
#define R_386_GOTPC     10
#define R_386_NUM 11

#define R_MIPS_NONE           0
#define R_MIPS_16       1
#define R_MIPS_32       2
#define R_MIPS_REL32          3
#define R_MIPS_26       4
#define R_MIPS_HI16           5
#define R_MIPS_LO16           6
#define R_MIPS_GPREL16        7
#define R_MIPS_LITERAL        8
```

EXHIBIT DD

Linux ELF Code

```
#define R_MIPS_GOT16          9
#define R_MIPS_PC16           10
#define R_MIPS_CALL16         11
#define R_MIPS_GPREL32        12
/* The remaining relocs are defined on Irix, although they are not
   in the MIPS ELF ABI.  */
#define R_MIPS_UNUSED1        13
#define R_MIPS_UNUSED2        14
#define R_MIPS_UNUSED3        15
#define R_MIPS_SHIFT5         16
#define R_MIPS_SHIFT6         17
#define R_MIPS_64       18
#define R_MIPS_GOT_DISP       19
#define R_MIPS_GOT_PAGE       20
#define R_MIPS_GOT_OFST       21
/*
 * The following two relocation types are specified in the MIPS ABI
 * conformance guide version 1.2 but not yet in the psABI.
 */
#define R_MIPS_GOTHI16        22
#define R_MIPS_GOTLO16        23
#define R_MIPS_SUB            24
#define R_MIPS_INSERT_A       25
#define R_MIPS_INSERT_B       26
#define R_MIPS_DELETE         27
#define R_MIPS_HIGHER         28
#define R_MIPS_HIGHEST        29
/*
 * The following two relocation types are specified in the MIPS ABI
 * conformance guide version 1.2 but not yet in the psABI.
 */
#define R_MIPS_CALLHI16       30
#define R_MIPS_CALLLO16       31
/*
 * This range is reserved for vendor specific relocations.
 */
#define R_MIPS_LOVENDOR       100
#define R_MIPS_HIVENDOR       127


/*
 * Sparc ELF relocation types
 */
#define     R_SPARC_NONE            0
#define     R_SPARC_8         1
#define     R_SPARC_16        2
#define     R_SPARC_32        3
#define     R_SPARC_DISP8           4
#define     R_SPARC_DISP16          5
#define     R_SPARC_DISP32          6
#define     R_SPARC_WDISP30         7
#define     R_SPARC_WDISP22         8
#define     R_SPARC_HI22            9
#define     R_SPARC_22        10
#define     R_SPARC_13        11
```

EXHIBIT DD

Linux ELF Code

```
#define     R_SPARC_LO10            12
#define     R_SPARC_GOT10           13
#define     R_SPARC_GOT13           14
#define     R_SPARC_GOT22           15
#define     R_SPARC_PC10            16
#define     R_SPARC_PC22            17
#define     R_SPARC_WPLT30          18
#define     R_SPARC_COPY            19
#define     R_SPARC_GLOB_DAT  20
#define     R_SPARC_JMP_SLOT  21
#define     R_SPARC_RELATIVE  22
#define     R_SPARC_UA32            23
#define R_SPARC_PLT32         24
#define R_SPARC_HIPLT22       25
#define R_SPARC_LOPLT10       26
#define R_SPARC_PCPLT32       27
#define R_SPARC_PCPLT22       28
#define R_SPARC_PCPLT10       29
#define R_SPARC_10            30
#define R_SPARC_11            31
#define R_SPARC_WDISP16       40
#define R_SPARC_WDISP19       41
#define R_SPARC_7       43
#define R_SPARC_5       44
#define R_SPARC_6       45

/* Bits present in AT_HWCAP, primarily for Sparc32.  */

#define HWCAP_SPARC_FLUSH       1    /* CPU supports flush instruction. */
#define HWCAP_SPARC_STBAR       2
#define HWCAP_SPARC_SWAP        4
#define HWCAP_SPARC_MULDIV      8
#define HWCAP_SPARC_V9        16
#define HWCAP_SPARC_ULTRA3    32

/*
 * 68k ELF relocation types
 */
#define R_68K_NONE      0
#define R_68K_32  1
#define R_68K_16  2
#define R_68K_8         3
#define R_68K_PC32      4
#define R_68K_PC16      5
#define R_68K_PC8 6
#define R_68K_GOT32     7
#define R_68K_GOT16     8
#define R_68K_GOT8      9
#define R_68K_GOT32O    10
#define R_68K_GOT16O    11
#define R_68K_GOT8O     12
#define R_68K_PLT32     13
#define R_68K_PLT16     14
#define R_68K_PLT8      15
#define R_68K_PLT32O    16
```

EXHIBIT DD

Linux ELF Code

```
#define R_68K_PLT16O    17
#define R_68K_PLT8O     18
#define R_68K_COPY      19
#define R_68K_GLOB_DAT  20
#define R_68K_JMP_SLOT  21
#define R_68K_RELATIVE  22

/*
 * Alpha ELF relocation types
 */
#define R_ALPHA_NONE            0       /* No reloc */
#define R_ALPHA_REFLONG         1       /* Direct 32 bit */
#define R_ALPHA_REFQUAD         2       /* Direct 64 bit */
#define R_ALPHA_GPREL32         3       /* GP relative 32 bit */
#define R_ALPHA_LITERAL         4       /* GP relative 16 bit w/optimization */
#define R_ALPHA_LITUSE          5       /* Optimization hint for LITERAL */
#define R_ALPHA_GPDISP          6       /* Add displacement to GP */
#define R_ALPHA_BRADDR          7       /* PC+4 relative 23 bit shifted */
#define R_ALPHA_HINT            8       /* PC+4 relative 16 bit shifted */
#define R_ALPHA_SREL16          9       /* PC relative 16 bit */
#define R_ALPHA_SREL32          10      /* PC relative 32 bit */
#define R_ALPHA_SREL64          11      /* PC relative 64 bit */
#define R_ALPHA_OP_PUSH         12      /* OP stack push */
#define R_ALPHA_OP_STORE        13      /* OP stack pop and store */
#define R_ALPHA_OP_PSUB         14      /* OP stack subtract */
#define R_ALPHA_OP_PRSHIFT      15      /* OP stack right shift */
#define R_ALPHA_GPVALUE         16
#define R_ALPHA_GPRELHIGH       17
#define R_ALPHA_GPRELLOW        18
#define R_ALPHA_IMMED_GP_16     19
#define R_ALPHA_IMMED_GP_HI32   20
#define R_ALPHA_IMMED_SCN_HI32  21
#define R_ALPHA_IMMED_BR_HI32   22
#define R_ALPHA_IMMED_LO32      23
#define R_ALPHA_COPY            24      /* Copy symbol at runtime */
#define R_ALPHA_GLOB_DAT        25      /* Create GOT entry */
#define R_ALPHA_JMP_SLOT        26      /* Create PLT entry */
#define R_ALPHA_RELATIVE        27      /* Adjust by program base */

/* Legal values for e_flags field of Elf64_Ehdr.  */

#define EF_ALPHA_32BIT        1     /* All addresses are below 2GB */


typedef struct elf32_rel {
  Elf32_Addr      r_offset;
  Elf32_Word      r_info;
} Elf32_Rel;

typedef struct elf64_rel {
  Elf64_Addr r_offset;  /* Location at which to apply the action */
  Elf64_Xword r_info;   /* index and type of relocation */
} Elf64_Rel;

typedef struct elf32_rela{
```

EXHIBIT DD

Linux ELF Code

```
  Elf32_Addr      r_offset;
  Elf32_Word      r_info;
  Elf32_Sword     r_addend;
} Elf32_Rela;

typedef struct elf64_rela {
  Elf64_Addr r_offset;  /* Location at which to apply the action */
  Elf64_Xword r_info;   /* index and type of relocation */
  Elf64_Sxword r_addend;      /* Constant addend used to compute value */
} Elf64_Rela;

typedef struct elf32_sym{
  Elf32_Word      st_name;
  Elf32_Addr      st_value;
  Elf32_Word      st_size;
  unsigned char   st_info;
  unsigned char   st_other;
  Elf32_Half      st_shndx;
} Elf32_Sym;

typedef struct elf64_sym {
  Elf64_Word st_name;         /* Symbol name, index in string tbl */
  unsigned char   st_info;    /* Type and binding attributes */
  unsigned char   st_other;   /* No defined meaning, 0 */
  Elf64_Half st_shndx;        /* Associated section index */
  Elf64_Addr st_value;        /* Value of the symbol */
  Elf64_Xword st_size;        /* Associated symbol size */
} Elf64_Sym;


#define EI_NIDENT 16

typedef struct elf32_hdr{
  unsigned char   e_ident[EI_NIDENT];
  Elf32_Half      e_type;
  Elf32_Half      e_machine;
  Elf32_Word      e_version;
  Elf32_Addr      e_entry;  /* Entry point */
  Elf32_Off e_phoff;
  Elf32_Off e_shoff;
  Elf32_Word      e_flags;
  Elf32_Half      e_ehsize;
  Elf32_Half      e_phentsize;
  Elf32_Half      e_phnum;
  Elf32_Half      e_shentsize;
  Elf32_Half      e_shnum;
  Elf32_Half      e_shstrndx;
} Elf32_Ehdr;

typedef struct elf64_hdr {
  unsigned char   e_ident[16];            /* ELF "magic number" */
  Elf64_Half e_type;
  Elf64_Half e_machine;
  Elf64_Word e_version;
  Elf64_Addr e_entry;         /* Entry point virtual address */
```

EXHIBIT DD

Linux ELF Code

```
  Elf64_Off e_phoff;          /* Program header table file offset */
  Elf64_Off e_shoff;          /* Section header table file offset */
  Elf64_Word e_flags;
  Elf64_Half e_ehsize;
  Elf64_Half e_phentsize;
  Elf64_Half e_phnum;
  Elf64_Half e_shentsize;
  Elf64_Half e_shnum;
  Elf64_Half e_shstrndx;
} Elf64_Ehdr;

/* These constants define the permissions on sections in the program
   header, p_flags. */
#define PF_R            0x4
#define PF_W            0x2
#define PF_X            0x1

typedef struct elf32_phdr{
  Elf32_Word      p_type;
  Elf32_Off p_offset;
  Elf32_Addr      p_vaddr;
  Elf32_Addr      p_paddr;
  Elf32_Word      p_filesz;
  Elf32_Word      p_memsz;
  Elf32_Word      p_flags;
  Elf32_Word      p_align;
} Elf32_Phdr;

typedef struct elf64_phdr {
  Elf64_Word p_type;
  Elf64_Word p_flags;
  Elf64_Off p_offset;         /* Segment file offset */
  Elf64_Addr p_vaddr;         /* Segment virtual address */
  Elf64_Addr p_paddr;         /* Segment physical address */
  Elf64_Xword p_filesz;       /* Segment size in file */
  Elf64_Xword p_memsz;        /* Segment size in memory */
  Elf64_Xword p_align;        /* Segment alignment, file & memory */
} Elf64_Phdr;

/* sh_type */
#define SHT_NULL  0
#define SHT_PROGBITS    1
#define SHT_SYMTAB      2
#define SHT_STRTAB      3
#define SHT_RELA  4
#define SHT_HASH  5
#define SHT_DYNAMIC     6
#define SHT_NOTE  7
#define SHT_NOBITS      8
#define SHT_REL         9
#define SHT_SHLIB 10
#define SHT_DYNSYM      11
#define SHT_NUM         12
#define SHT_LOPROC      0x70000000
#define SHT_HIPROC      0x7fffffff
```

EXHIBIT DD

Linux ELF Code

```
#define SHT_LOUSER      0x80000000
#define SHT_HIUSER      0xffffffff
#define SHT_MIPS_LIST           0x70000000
#define SHT_MIPS_CONFLICT       0x70000002
#define SHT_MIPS_GPTAB          0x70000003
#define SHT_MIPS_UCODE          0x70000004

/* sh_flags */
#define SHF_WRITE 0x1
#define SHF_ALLOC 0x2
#define SHF_EXECINSTR   0x4
#define SHF_MASKPROC    0xf0000000
#define SHF_MIPS_GPREL  0x10000000

/* special section indexes */
#define SHN_UNDEF 0
#define SHN_LORESERVE   0xff00
#define SHN_LOPROC      0xff00
#define SHN_HIPROC      0xff1f
#define SHN_ABS         0xfff1
#define SHN_COMMON      0xfff2
#define SHN_HIRESERVE   0xffff
#define SHN_MIPS_ACCOMON        0xff00

typedef struct {
  Elf32_Word      sh_name;
  Elf32_Word      sh_type;
  Elf32_Word      sh_flags;
  Elf32_Addr      sh_addr;
  Elf32_Off sh_offset;
  Elf32_Word      sh_size;
  Elf32_Word      sh_link;
  Elf32_Word      sh_info;
  Elf32_Word      sh_addralign;
  Elf32_Word      sh_entsize;
} Elf32_Shdr;

typedef struct elf64_shdr {
  Elf64_Word sh_name;         /* Section name, index in string tbl */
  Elf64_Word sh_type;         /* Type of section */
  Elf64_Xword sh_flags;       /* Miscellaneous section attributes */
  Elf64_Addr sh_addr;         /* Section virtual addr at execution */
  Elf64_Off sh_offset;        /* Section file offset */
  Elf64_Xword sh_size;        /* Size of section in bytes */
  Elf64_Word sh_link;         /* Index of another section */
  Elf64_Word sh_info;         /* Additional section information */
  Elf64_Xword sh_addralign;   /* Section alignment */
  Elf64_Xword sh_entsize;     /* Entry size if section holds table */
} Elf64_Shdr;

#define     EI_MAG0           0           /* e_ident[] indexes */
#define     EI_MAG1           1
#define     EI_MAG2           2
#define     EI_MAG3           3
#define     EI_CLASS    4
```

EXHIBIT DD

Linux ELF Code

```
#define     EI_DATA           5
#define     EI_VERSION  6
#define     EI_PAD            7

#define     ELFMAG0           0x7f        /* EI_MAG */
#define     ELFMAG1           'E'
#define     ELFMAG2           'L'
#define     ELFMAG3           'F'
#define     ELFMAG            "\177ELF"
#define     SELFMAG           4

#define     ELFCLASSNONE      0           /* EI_CLASS */
#define     ELFCLASS32  1
#define     ELFCLASS64  2
#define     ELFCLASSNUM 3

#define ELFDATANONE     0           /* e_ident[EI_DATA] */
#define ELFDATA2LSB     1
#define ELFDATA2MSB     2

#define EV_NONE         0           /* e_version, EI_VERSION */
#define EV_CURRENT      1
#define EV_NUM          2

/* Notes used in ET_CORE */
#define NT_PRSTATUS     1
#define NT_PRFPREG      2
#define NT_PRPSINFO     3
#define NT_TASKSTRUCT   4
#define NT_PRFPXREG     20

/* Note header in a PT_NOTE section */
typedef struct elf32_note {
  Elf32_Word      n_namesz;   /* Name size */
  Elf32_Word      n_descsz;   /* Content size */
  Elf32_Word      n_type;           /* Content type */
} Elf32_Nhdr;

/* Note header in a PT_NOTE section */
typedef struct elf64_note {
  Elf64_Word n_namesz;  /* Name size */
  Elf64_Word n_descsz;  /* Content size */
  Elf64_Word n_type;    /* Content type */
} Elf64_Nhdr;

#if ELF_CLASS == ELFCLASS32

extern Elf32_Dyn _DYNAMIC [];
#define elfhdr          elf32_hdr
#define elf_phdr  elf32_phdr
#define elf_note  elf32_note

#else

extern Elf64_Dyn _DYNAMIC [];
```

EXHIBIT DD

Linux ELF Code

```
#define elfhdr          elf64_hdr
#define elf_phdr  elf64_phdr
#define elf_note  elf64_note

#endif


#endif /* _LINUX_ELF_H */
```

CERTIFICATE OF SERVICE

Plaintiff/Counterclaim Defendant, The SCO Group, Inc., hereby certifies that a true and correct copy of the foregoing was served on Defendant IBM on the 5th day of July, 2005 by U.S. Mail to:

David Marriott, Esq.
CRAVATH SWAINE & MOORE LLP
Worldwide Plaza
825 Eighth Avenue
New York, NY 10019

Donald Rosenberg, Esq.
1133 Westchester Avenue
White Plains, NY 10604

Todd Shaughnessy, Esq.
SNELL & WILMER LLP
1200 Gateway Tower West
15 West South Temple
Salt Lake City, UT 84101-1004

Laura K. Chaves